IRIA

CENTRE DE RENNES
IRISA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 584

DÉRIVATION DE PROTOCOLES D'EXCLUSION MUTUELLE ÉQUITABLES

René THORAVAL

Novembre 1986

DERIVATION DE PROTOCOLES D'EXCLUSION MUTUELLE EQUITABLES

DERIVATION OF STARVATION-FREE MUTUAL EXCLUSION PROTOCOLS

RENÉ THORAVAL

RESUME

Dans ce rapport , nous présentons une méthode de dérivation de protocoles parallèles. La dérivation d'un protocole vérifiant certaines propriétés est réalisée, à partir d'un protocole très simple vérifiant seulement une partie d'entre elles, à travers une succession de transformations élémentaires mettant progressivement en place des mécanismes abstraits de synchronisation. A l'issue de ces transformations, des représentations adaptées de ces mécanismes abstraits sont proposées.

Cette méthode de dérivation s'avère informelle mais nous évoquons la possibilité de la formaliser partiellement.

Elle convient à la construction de protocoles d'exclusion mutuelle mais nous pensons qu'elle pourrait être adaptée à la construction d'autres types de protocoles. Son emploi nous permet de construire plusieurs protocoles équitables d'exclusion mutuelle, retrouvant, parmi eux, certaines solutions classiques dont la comparaison se trouve facilitée.

ABSTRACT

In this report, a method for deriving parallel protocols is proposed. We derive a protocol that verifies some given properties from a very simple one that only verifies a part of them. This is done through a sequence of elementary transformations, each of them introducing an abstract synchronization mechanism. Such a sequence is followed by the choice of consistent representations of the abstract mechanisms.

This derivation method is informal but we show how it can be partially formalized.

It fits for mutual exclusion protocols but we think it could be tailored to other kinds of protocols. We use it to build several fair mutual exclusion protocols. By the way, we retrieve some well-known ones whose comparison is made easier.

TABLE DES MATIERES

DERIVATION DE PROTOCOLES D'EXCLUSION MUTUELLE EQUITABLES*

INTRODUCTION

La littérature présentant des protocoles parallèles distribués ou non est à présent abondante, en particulier celle concernant les protocoles d'exclusion mutuelle. Nombre d'entre eux sont présentés dans (Raynal 84) et (Raynal 85a).

Mais la plupart des protocoles présentés dans la littérature le sont, schématiquement, sous la forme suivante :

(I) description du protocole (variables utilisées, programmes des processus)

(2) énoncé des propriétés du protocole (exclusion mutuelle, équité...)

(3) preuve de la réalisation de ces propriétés par le protocole.

Ce mode de présentation laisse de nombreuses autres questions importantes sans réponse, par exemple :

- comment le protocole a-t-il été construit ?
- à partir de quelle(s) idée(s) abstraite(s) et intuitive(s) simple(s) l'a-t-il été ?
- pourquoi certaines décisions ont-elles été prises lors de sa construction (ordre des instructions, rôle de telle d'entre elles...) ? ces décisions ont-elles été essentielles à la réalisation des propriétés attendues ? ont-elles seulement visé à optimiser, en un certain sens, le protocole, ou simplement à en faciliter la compréhension ? quelles auraient été les conséquences de décisions légèrement différentes ?

En fait, ce mode de présentation oblige le lecteur exigeant à répondre par lui-même à ces questions. Il est alors contraint de déployer un effort de compréhension dont l'ampleur et la dispersion auraient pourtant pu être réduites si l'auteur lui avait livré les clés de son protocole. En particulier, il lui est difficile de se représenter, d'une façon intuitive et abstraite (et donc essentielle), les mécanismes mis en jeu et de voir d'emblée en quoi ce protocole diffère des autres. Bref, ce mode de présentation ne permet de comprendre ni facilement ni rapidement.

---

* Ce travail a été réalisé dans le cadre du pôle "Algorithmes distribués et Evaluation de performances" de C^3.

Cependant, par exemple :

- (Francez 8I) propose une méthode de construction de protocoles distribués et l'utilise pour retrouver notamment le protocole de détection de la terminaison de (Francez 79) ;
- (Bochmann 79) et (André 83) proposent une méthode, utilisée notamment dans (Verjus 86), pour construire des protocoles distribués à partir d'expressions abstraites définies en termes de compteurs de (Robert 77).

Dans le rapport qui suit, nous adoptons une démarche de construction (et d'exposé) de protocoles qui vise à répondre aux questions évoquées plus haut. Nous considérons ici des protocoles d'exclusion mutuelle mais une démarche analogue pourrait être appliquée à d'autres types de protocoles parallèles.

Nous sommes partis de l'intuition suivante :

-derrière une littérature abondante sur le problème de l'exclusion mutuelle, se trouvent seulement un nombre assez restreint de mécanismes de base, perceptibles dès lors que l'on parvient à saisir ces protocoles de façon abstraite ;
- la situation est aujourd'hui mûre pour mener à bien ce travail d'abstraction.

A partir d'un tel point de vue, nous avons étudié dans (Verjus 86) des protocoles équitables d'exclusion mutuelle par circulation d'un privilège sur un anneau virtuel et montré notamment, l'identité, d'un certain point de vue abstrait, des solutions de (Dijkstra 74) et (Raynal 85b).

Nous considérons ici un autre type de solutions au problème de l'exclusion mutuelle.

Nous définissons d'abord, au paragraphe I, le type de systèmes considérés et nous indiquons ce que nous entendons par dérivation.

Dans les paragraphes 2 et 3, nous définissons les bases à partir desquelles nous dériverons ultérieurement des protocoles équitables :

- au paragraphe 2, nous présentons un protocole abstrait d'exclusion mutuelle par translation d'un privilège sur un anneau virtuel ; ce paragraphe a pour seul objet d'introduire le mécanisme abstrait de translation que nous utiliserons par la suite pour dériver du protocole "germe", construit au paragraphe 3, des protocoles équitables ;
- au paragraphe 3, nous construisons ce germe ; plus précisément, partant d'un protocole (I) très simple, présenté à l'alinéa 3.I, assurant l'exclusion mutuelle mais comportant un risque d'interblocage, nous construisons, à l'alinéa 3.2, un protocole (2) (utilisé notamment par (Katseff 78)) assurant toujours l'exclusion mutuelle, cette fois sans risque d'interblocage ; ce germe (2) est cependant inéquitable.

Au paragraphe 4, nous dérivons, à partir de ce germe, grâce à deux transformations élémentaires, un protocole d'exclusion mutuelle équitable. La première transformation met en place le mécanisme abstrait de translation, la seconde un mécanisme décelé dans, puis abstrait de la solution classique de (Eisenberg 72).

Au paragraphe 5, nous proposons deux représentations de ces mécanismes abstraits.

Au paragraphe 6, nous présentons quelques variantes de ces mécanismes. Elles nous permettent de dériver du germe, d'autres protocoles équitables. Parmi eux, nous retrouvons certaines solutions classiques dont nous mettons en évidence les ressemblances et différences.

I. PRELIMINAIRES

Nous noterons, pour tout couple (x,y) d'éléments de l'ensemble $\{0,\ldots,n-1\}$, par $x \boxplus y$ (resp. $x \boxminus y$) le représentant dans cet ensemble de la classe modulo n de (x+y) (resp. (x-y)).

I.I. Système, processus, protocole, variables et communication

Nous considérons des systèmes parallèles formés de n processus, n étant un entier quelconque strictement supérieur à un. Ces systèmes sont de la forme :

phase initiale ; $[P_0 \parallel P_1 \parallel \ldots \parallel P_{n-1}]$

Au cours de la phase initiale, les variables qui doivent avoir une valeur initiale définie se voient attribuer cette valeur. A l'issue de cette phase, les n processus sont activés en parallèle.

A l'issue de la phase initiale, nous disons qu'un processus est actif (resp. passif) à un instant donné s'il est en cours (resp. n'est pas en cours) d'exécution de son protocole à cet instant.

Lorsqu'il est actif, un processus quelconque a une vitesse d'exécution finie et non nulle, mais quelconque.

Dans le système du paragraphe 2 d'une part, et dans les systèmes des paragraphes suivants d'autre part, les processus ont un comportement différent :

- au paragraphe 2, tout processus passif à un instant donné (re-)devient actif au bout d'un temps fini ;
- aux paragraphes suivants, tout processus passif à un instant donné peut tout aussi bien (re-)devenir actif au bout d'un temps fini que demeurer définitivement passif.

Dans chacun des systèmes considérés, les programmes des protocoles de tous les processus sont syntaxiquement identiques. Ils ne se distinguent que par référence aux numéros d'identification des processus. Notons qu'une telle identité syntaxique n'implique généralement pas une identité de comportement.

Nous considérons trois types de variables :

- des variables locales à un processus : seul celui-ci peut les consulter et les mettre à jour ;
- des variables de communication appartenant à un processus : seul celui-ci peut les mettre à jour, ses concurrents pouvant seulement les consulter ;
- des variables partagées : tous les processus peuvent les consulter et les mettre à jour.

Nous appelons système distribué un système où n'existe pas de variable partagée. La communication entre processus y est réalisée à l'aide de variables de communication.

I.2. Equité

Nous entendons par protocole équitable (cf. (Raynal 84)) un protocole dans lequel tout processus candidat à sa section critique peut l'exécuter au bout d'un temps fini. C'est donc un protocole dans lequel un processus candidat quelconque ne peut être réduit à l'état de famine.

De plus, dans la plupart des protocoles équitables présentés, l'attente d'un processus candidat quelconque est bornée, non pas (bien sûr) en termes de temps, mais de nombre de tours (ou dépassements) (cf. (Raynal 84)). Nous parlons alors d'"attente maximale d'un processus candidat". Pour un processus candidat quelconque, c'est le nombre maximal, pour toutes les histoires possibles du système, d'exécutions complètes de leur protocole par l'ensemble de ses concurrents, pendant un certain intervalle de temps allant de sa déclaration de candidature à son entrée en section critique.

I.3. Opérations atomiques

Dans les systèmes considérés, les seules opérations atomiques sont la lecture et la mise à jour de la valeur d'une variable. Ce sont donc les seules opérations pour lesquelles nous supposons a priori que leur exécution simultanée a les mêmes effets qu'une exécution en série.

I.4. Schéma d'exécution des instructions complexes

Dans les programmes présentés, nous utilisons des instructions complexes du type suivant : (si $COND_i$ alors ... fsi), (si $COND_i$ alors ... sinon ... fsi), (tant que $COND_i$ faire ... ftq), (répéter ... jusqu'à $COND_i$), (attendre $COND_i$). L'instruction (attendre $COND_i$) représente une attente active : (répéter (instruction vide) jusqu'à $COND_i$). L'expression logique $COND_i$ porte sur un ensemble V_i de variables contenant :

- des variables de communication du processus P_i et de ses concurrents ;
- la variable partagée existant éventuellement dans le système.

Plutôt que d'imposer un ordre particulier aux opérations menant à l'évaluation d'une expression $COND_i$, nous suivons (Pnueli 84) en adoptant un schéma plus souple : le processus P_i effectue ces différentes opérations dans un ordre quelconque, défini par exemple, à chaque évaluation, par tirage au sort. Ce schéma est également adopté lors de l'évaluation des conditions complexes du type présenté ci-dessous. Nous faisons ce choix pour bien montrer que l'ordre choisi pour ces opérations n'est pas décisif quant au bon fonctionnement des protocoles présentés.

Nous considérons, entre autres, des conditions complexes du type suivant : (quelqu'un de E_i dans val), (personne de E_i dans val), (tous de E_i dans val) où E_i est un sous-ensemble de $\{0,\ldots,n-1\}$, variant éventuellement au cours de l'histoire du système considéré, et val un sous-ensemble de VAL, VAL étant l'ensemble des valeurs que peuvent prendre les variables de communication $ETAT_j$ des processus P_j (j entier quelconque de l'ensemble $\{0,\ldots,n-1\}$). Nous généralisons ainsi les expressions (quelqu'un d'autre dans val), (personne d'autre dans val), (tous les autres dans val) employées par (Pnueli 84). Ainsi, par exemple, l'expression (quelqu'un d'autre dans val) est représentée par (quelqu'un de $\{0,\ldots,n-1\} - \{i\}$ dans val). L'évaluation d'une expression $COND_i$ comme, par exemple, (quelqu'un de E_i dans val) est réalisée selon le schéma d'exécution suivant :

```
Evaluer E_i      / inutile quand l'ensemble E_i a une valeur statique,
                   par exemple : E_i = {0,...,n-I} - {i} /
RESTE A PARCOURIR_i := E_i
QUELQU'UN_i := faux
tant que RESTE A PARCOURIR_i ≠ ∅ et non QUELQU'UN_i faire
    J_i := un élément quelconque de RESTE A PARCOURIR_i
    si ETAT_Ji dans val
        alors QUELQU'UN_i := vrai
        sinon Enlever (J_i, RESTE A PARCOURIR_i)
    fsi
ftq
```

où RESTE A PARCOURIR$_i$, QUELQU'UN$_i$, J_i sont des variables locales au processus P_i, la première de type ensemble et à valeurs dans l'ensemble $P(\{0,\ldots,n-I\})$, la seconde de type logique, la dernière à valeurs dans l'ensemble $\{0,\ldots,n-I\}$, toutes de valeurs initiales quelconques. L'expression COND$_i$ prend alors la valeur finale de la variable logique QUELQU'UN$_i$.

Nous considérons des conditions complexes de ce type pour simplifier, par la suite, la lecture des protocoles présentés.

I.5. Dérivation

Lorsque nous parlons de dérivation d'un protocole (B) à partir d'un protocole (A), nous entendons adjonction au protocole (A) d'un mécanisme de synchronisation :

- préservant certaines propriétés, jugées intéressantes, du protocole (A) (exemple : exclusion mutuelle et absence de risque d'interblocage),
- dotant le protocole (B) d'une propriété que ne possède pas le protocole (A) (exemple : absence de famine pour tout processus).

Le mode de dérivation envisagé s'avère informel. Cependant, comme nous le montrons en Annexe, il est possible de le formaliser, du moins partiellement.

2. EXCLUSION MUTUELLE PAR TRANSLATION PAS A PAS D'UN PRIVILEGE SUR UN ANNEAU VIRTUEL

Dans ce paragraphe, rappelons-le, tout processus passif à un instant donné (re-)devient actif au bout d'un temps fini.

Considérons le protocole abstrait suivant dans lequel un processus P_i (i entier quelconque de l'ensemble $\{0,\ldots,n-I\}$) est prioritaire (autrement dit détient le privilège de pouvoir entrer en section critique) si et seulement si (JE SUIS PRIORITAIRE$_i$ = vrai) :

```
attendre JE SUIS PRIORITAIRE_i
SC_i
TRANSLATER PRIORITE DE UN_i
```

où :

- initialement, le processus P_0 est le seul processus prioritaire ;
- pour tout entier i de l'ensemble $\{0,\ldots,n-I\}$, à l'instant de commencement d'une exécution de la procédure TRANSLATER PRIORITE DE UN$_i$, le processus P_i est seul prioritaire et le demeure jusqu'à la fin de cette exécution, le processus $P_{i \oplus I}$ devenant alors, à son tour, seul prioritaire ; autrement dit, cette exécution par le processus P_i a pour effet de translater le privilège d'un pas sur l'anneau virtuel où tout processus P_i a pour successeur le processus $P_{i \oplus I}$.

Ce protocole est abstrait au sens où il fait abstraction de la représentation des relations de priorité entre processus et de celle de leur évolution. Selon la représentation choisie (cf. paragraphe 5), nous pouvons dériver de ce protocole abstrait un protocole utilisant une variable partagée ou un protocole distribué.

Il est clair que ce protocole abstrait assure l'exclusion mutuelle sur (SC_i) (à tout instant, un processus et un seul est prioritaire) de manière équitable (tôt ou tard, chaque processus candidat devient prioritaire puisque tout processus passif à un instant donné (re-)devient actif au bout d'un temps fini).

3. CONSTRUCTION DU PROTOCOLE GERME

Au paragraphe précédent, nous avons considéré des processus qui, lorsqu'ils sont passifs à un instant donné, (re-)deviennent actifs au bout d'un temps fini. A partir de maintenant, nous affaiblissons cette hypothèse : tout processus passif à un instant donné peut tout aussi bien (re-)devenir actif au bout d'un temps fini que demeurer définitivement passif.

Sous ces nouvelles hypothèses, avant de construire des protocoles équitables, nous allons, dans ce paragraphe 3, construire un protocole (2) d'exclusion mutuelle sans risque d'interblocage, mais inéquitable. C'est à partir de ce protocole germe que nous dériverons, dans les paragraphes suivants, des protocoles équitables, en utilisant notamment le procédé de translation présenté au paragraphe 2.

3.I. Un protocole d'exclusion mutuelle comportant un risque d'interblocage

Supposons donné le protocole distribué suivant (I) :

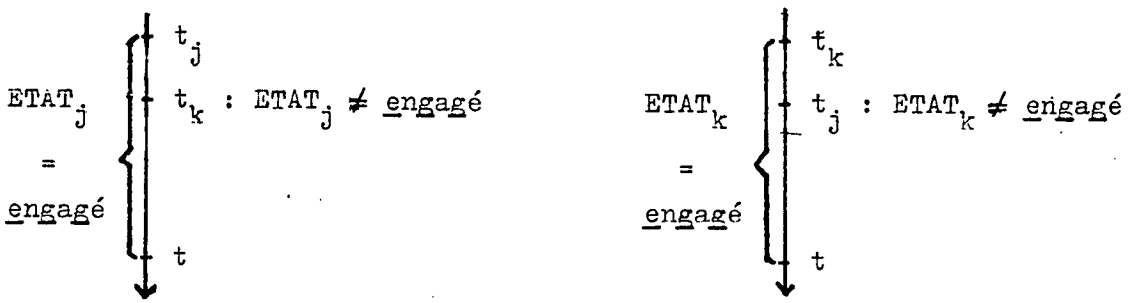
- où chaque processus P_i dispose d'une variable de communication $ETAT_i$ à valeurs dans l'ensemble {dehors, en_attente, engagé} et de valeur initiale dehors (la valeur de cette variable représente l'état de participation du processus P_i à la compétition) ;
- dont le programme est :

répéter
  $ETAT_i$:= en_attente
  $ETAT_i$:= engagé
jusqu'à (personne d'autre dans {engagé})
SC_i
$ETAT_i$:= dehors

Ce protocole que nous supposons donné peut, en réalité, être dérivé à partir d'une expression abstraite de synchronisation portant sur des compteurs de (Robert 77), expression impliquant la propriété d'exclusion mutuelle (voir (Thoraval 87)).

Ce protocole assure l'exécution de l'action (SC_i) en exclusion mutuelle. En fait, le processus P_i bénéficiaire de l'exclusion mutuelle en dispose dès l'instant de terminaison de l'évaluation favorable de l'expression (personne d'autre dans {engagé}) et ne la relâche qu'à l'instant où il termine la modification de la variable $ETAT_i$.

Démontrons-le en raisonnant par l'absurde. Considérons un hypothétique instant t auquel l'exclusion mutuelle est violée. A cet instant t, il existe donc deux processus distincts P_j et P_k en cours d'exécution respectivement de (SC_j) et (SC_k). Lors de sa dernière évaluation, avant l'instant t, de l'expression (personne d'autre dans {engagé}), le processus P_j (resp. P_k) a lu la valeur de la variable $ETAT_k$ (resp. $ETAT_j$) et cette lecture s'est terminée à un instant t_j (resp. t_k). La présence du processus P_j en section critique à l'instant t implique qu'à l'instant t_j : $ETAT_k \neq$ engagé et $ETAT_j$ = engagé. De même, la présence du processus P_k en section critique à l'instant t implique qu'à l'instant t_k : $ETAT_k$ = engagé et $ETAT_j \neq$ engagé. Il en résulte, tout d'abord, que les instants t_j et t_k sont distincts. D'autre part, sur l'intervalle $[t_j, t]$ (resp. $[t_k, t]$), la valeur de la variable $ETAT_j$ (resp. $ETAT_k$) demeurant constamment engagé, il en résulte que nous ne pouvons avoir ni ($t_j < t_k$) ni ($t_j > t_k$), comme le montre la figure suivante :



Puisque nous ne pouvons avoir ni ($t_j = t_k$) ni ($t_j < t_k$) ni ($t_j > t_k$), il s'avère qu'il ne peut exister d'instant t auquel l'exclusion mutuelle est violée.

Malheureusement, le protocole (I) comporte un risque d'interblocage. Par exemple, deux processus exécutant indéfiniment, à tour de rôle, une instruction de leur protocole, se trouvent en situation d'interblocage. L'existence d'un tel risque d'interblocage n'a d'ailleurs rien d'étonnant. Le protocole (I) peut être, en effet, réécrit sous forme "fortement symétrique" au sens de Burns, et il est démontré dans (Burns 81) qu'il ne peut exister de protocole distribué fortement symétrique assurant l'exclusion mutuelle sans risque d'interblocage.

Intuitivement, en effet, si les processus ne se différencient pas spontanément par leurs vitesses d'exécution, ils n'ont alors aucun autre moyen de le faire.

3.2. Dérivation du protocole germe ne comportant plus de risque d'interblocage

Le protocole (I) comporte un risque d'interblocage. Mais si, spontanément, les actions des différents processus se sérialisent correctement, l'interblocage ne se produit pas. Pour construire, à partir de ce protocole, un protocole (2) d'exclusion mutuelle sans risque d'interblocage, il reste alors à trouver un moyen de forcer, si nécessaire, en cas de conflit, cette sérialisation, au bout d'un temps fini. Il suffit pour cela de doter les processus d'un mécanisme d'élection fonctionnant quelles que soient leurs vitesses d'exécution. Alors, en cas de conflit, au bout d'un temps fini :

- l'un des concurrents se singularise,
- et cette singularité est reconnue par tous.

La symétrie du protocole (I) est ainsi brisée. Divers mécanismes d'élection sont connus. Nous allons reconstruire l'un d'entre eux.

Pour cela, partons d'une situation d'interblocage du protocole (I). Dans une telle situation, au bout d'un temps fini :

- il existe au moins deux processus bloqués en attente active,
- la situation se stabilise au sens où plus aucun processus n'entre par la suite dans cette phase-là.

A partir de ce moment-là, il existe, parmi les concurrents, un processus P_{iO} dont le numéro d'identification est le plus petit. Les différents concurrents ont tout loisir, au cours de leurs cycles infinis, de s'en rendre compte. Pourquoi donc ne pas se servir de cette connaissance accessible à chacun d'entre eux pour sortir de cette situation inextricable ? Il suffit que chacun annule son engagement dès qu'il se rend compte qu'il y a plus prioritaire que lui et attende alors sagement de devenir, peut-être un jour, le plus prioritaire. Ce procédé est très voisin, quant à la prévention de l'interblocage, de celui utilisé dans le "wait-die system" de (Rosenkrantz 78).

Nous obtenons alors le protocole distribué (2) suivant (dérivé du protocole (I) par simple insertion d'une instruction d'attente active adéquate, brisant la symétrie du protocole (I)) :

- où chaque processus P_i dispose encore d'une variable de communication $ETAT_i$ à valeurs dans l'ensemble {dehors, en_attente, engagé} et de valeur initiale dehors ;

- dont le programme est :

répéter
    $ETAT_i$:= en_attente
    attendre (tous de PRIORITAIRES SUR SOI dans {dehors})
    $ETAT_i$:= engagé
jusqu'à (personne d'autre dans {engagé})
SC_i
$ETAT_i$:= dehors

Pour un processus P_i quelconque, la valeur de l'ensemble PRIORITAIRES SUR SOI est invariante au cours de l'histoire du système : c'est toujours {0,...,i-1}. Il est donc inutile au processus P_i d'évaluer à chaque fois cet ensemble (cf. alinéa I.4).

Ce protocole (2) est donné dans (Katseff 78) et utilisé de fait dans (Knuth 66), (de Bruijn 68) et (Eisenberg 72).

L'insertion de l'instruction d'attente active (attendre (tous de PRIORITAIRES SUR SOI dans {dehors}) dans le protocole (I) correspond à l'adjonction à ce protocole d'un mécanisme de synchronisation (cf. alinéa I.5) qui, tout en préservant la propriété d'exclusion mutuelle (cf. Annexe), permet, par construction, de supprimer tout risque d'interblocage.

Malheureusement ce protocole (2) n'est pas équitable. Les relations de priorité entre processus y sont, en effet, figées (si nous notons par "$P_j > P_k$" le fait que P_j soit prioritaire sur P_k, la relation ($P_0 > P_1 > P_2 > \ldots > P_{n-1}$) est un invariant du système). Il est facile de construire un scénario dans lequel un processus, autre que le processus P_0, est en état de famine. Par exemple, le processus P_0 peut réduire à l'état de famine le processus P_1 en se portant sans cesse et très rapidement candidat à sa section critique. Il suffit que le processus P_1 consulte indéfiniment au mauvais moment la variable $ETAT_0$ i.e. lorsqu'elle a la valeur engagé.

Dans les paragraphes suivants, nous allons dériver de ce protocole-germe (2) des protocoles équitables. A cette fin, nous rendrons évolutives les relations de priorité entre processus.

4. DERIVATION D'UN PROTOCOLE ABSTRAIT EQUITABLE A PARTIR DU GERME

Les deux protocoles présentés dans ce paragraphe sont des protocoles abstraits au sens défini au paragraphe 2.

4.I. Une dérivation en deux étapes

Le germe est inéquitable, pour le moins parce que les relations de priorité entre processus y sont invariantes.

La dérivation d'un protocole équitable, à partir de ce germe, va alors s'effectuer en deux étapes.

La première étape consistera à adjoindre au germe un mécanisme de synchronisation (cf. alinéa I.5) faisant évoluer de manière équitable les relations de priorité entre processus, sans toutefois altérer les propriétés intéressantes du germe (exclusion mutuelle, absence de risque d'interblocage). Qu'entendons-nous par évolution équitable des relations de priorité ? Considérant l'ensemble C(t) des passages en section critique des processus du système après un certain instant t, une évolution équitable des relations de priorité est une évolution telle que, pour tout processus P_i (i entier quelconque de l'ensemble $\{0,\ldots,n-1\}$) et toute histoire du système où l'ensemble C(t) est infini, le processus P_i devienne une infinité de fois prioritaire après l'instant t. L'adjonction de ce mécanisme permettra donc d'offrir régulièrement, à tout processus candidat, une chance sérieuse d'entrer en section critique. Toutefois, cette adjonction s'avérera insuffisante pour assurer l'équité du protocole (3) résultant. En effet, un processus candidat quelconque pourra, victime d'une coalition de ses (n-I) concurrents et réagissant toujours à contre-temps, laisser passer indéfiniment cette chance et être ainsi réduit à l'état de famine.

La deuxième étape de dérivation consistera alors à adjoindre au protocole (3) un mécanisme de synchronisation (cf. alinéa I.5) capable d'empêcher qu'un processus candidat quelconque soit victime d'une telle coalition de ses concurrents. L'adjonction de ce mécanisme préservera les propriétés intéressantes du protocole (3) : exclusion mutuelle, absence de risque d'interblocage, évolution équitable des relations de priorité.

4.2. Première étape de la dérivation : translation pas à pas des relations de priorité

4.2.I. Un mécanisme de translation assurant l'évolution équitable des relations de priorité

Le germe suppose qu'un processus quelconque puisse communiquer avec l'un quelconque de ses concurrents (maillage complet). Considérons, sur la base de ce maillage complet, l'anneau virtuel lié à la numérotation des processus,

anneau sur lequel le successeur d'un processus P_i quelconque est le processus $P_{i \boxplus I}$ (cf. paragraphe 2).

Pour assurer une évolution équitable des relations de priorité, une solution simple consiste à confier à tout processus P_i sortant de section critique, le soin de modifier, avant de relâcher l'exclusion mutuelle, les relations de priorité par l'exécution d'une procédure TRANSLATER PRIORITE DE UN_i définie comme au paragraphe 2. Ainsi, si à l'instant de commencement de cette exécution, un certain processus P_k est seul prioritaire (notons qu'ici, contrairement au paragraphe 2, nous n'avons plus nécessairement : i = k), alors il le demeure jusqu'à la fin de cette exécution, moment auquel le processus $P_{k \boxplus I}$ se retrouve seul prioritaire. Par là-même, toutes les relations de priorité entre processus sont translatés d'un pas le long de l'anneau virtuel. Nous passons ainsi de $(P_k > P_{k \boxplus I} > P_{k \boxplus 2} > \cdots > P_{k \boxminus I})$ à $(P_{k \boxplus I} > P_{k \boxplus 2} > \cdots > P_{k \boxminus I} > P_k)$.

L'adjonction au germe du mécanisme de synchronisation correspondant fait donc que les relations de priorité peuvent évoluer au cours de l'histoire du système, y compris au cours de l'évaluation par un processus P_i quelconque de l'expression (tous de PRIORITAIRES SUR SOI dans {dehors}). Ainsi, le processus P_i peut juger, en commençant l'évaluation de cette expression (cf. alinéa I.4), qu'un certain processus P_k est prioritaire. Au cours de cette évaluation, il se peut, par exemple, qu'un processus P_j, sortant de section critique, translate d'un pas les relations de priorité, rendant ainsi prioritaire le processus $P_{k \boxplus I}$.

Pour souligner cette possibilité, nous remplacerons par la suite, dans le germe, l'expression (tous de PRIORITAIRES SUR SOI dans {dehors}) par l'expression (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors}).

4.2.2. Obtention d'un protocole par composition de ce mécanisme avec le germe

Le protocole (3) suivant est dérivé du germe par simple insertion de l'action TRANSLATER PRIORITE DE UN_i. Pour ce protocole :

- chaque processus P_i dispose encore d'une variable $ETAT_i$ à valeurs dans l'ensemble {dehors, en_attente, engagé} et de valeur initiale dehors ;
- initialement, le processus P_0 est seul prioritaire ;
- le programme exécuté est le suivant :

```
répéter
    ETAT_i := en_attente
    attendre (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors})
    ETAT_i := engagé
jusqu'à (personne d'autre dans {engagé})
SC_i
TRANSLATER PRIORITE DE UN_i
ETAT_i := dehors
```

Pour chaque évaluation de l'expression (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors}), tout processus P_i évalue d'abord l'ensemble SUPPOSES PRIORITAIRES SUR SOI (cf. alinéa I.4). Soit PRIOR l'ensemble des variables dont les valeurs instantanées déterminent l'état instantané des relations de priorité (selon la représentation choisie - voir paragraphe 5 - PRIOR contient seulement une variable partagée ou bien contient n variables de communication). Pour évaluer l'ensemble SUPPOSES PRIORITAIRES SUR SOI, le processus P_i évalue d'abord, dans un ordre quelconque, les variables de l'ensemble PRIOR. Puis s'il estime, sur la base de ces valeurs, qu'un certain processus P_k est prioritaire, il en déduit :

- si $k \leqslant i$: SUPPOSES PRIORITAIRES SUR SOI = $\{k,\ldots,i-1\}$,
- si $k > i$: SUPPOSES PRIORITAIRES SUR SOI = $\{0,\ldots,i-1\} \cup \{k,\ldots,n-1\}$.

La dérivation du protocole (3) assure l'héritage des propriétés intéressantes du germe : exclusion mutuelle sans risque d'interblocage (cf. Annexe). Notons qu'ici encore, un processus P_i bénéficiaire de l'exclusion mutuelle en dispose dès l'instant de terminaison de l'évaluation favorable de l'expression (personne d'autre dans {engagé}) et ne la relâche que lorsqu'il termine la modification de la variable $ETAT_i$. Il modifie donc les relations de priorité alors qu'il dispose de l'exclusion mutuelle.

Notons enfin que l'on peut voir ce protocole (3) comme le résultat d'une transformation du protocole du paragraphe 2 (translation pas à pas d'un privilège sur un anneau virtuel), transformation contribuant à adapter ce dernier au cas où n'importe quel processus passif à un instant donné peut le demeurer définitivement.

4.2.3. Le protocole obtenu demeure inéquitable

Malheureusement, sauf dans le cas de deux processus (n = 2), le protocole (3) est inéquitable : un processus candidat quelconque (y compris P_0) peut être réduit à l'état de famine.

En effet, si un processus est candidat, ses (n-I) concurrents peuvent se coaliser pour lui interdire définitivement l'accès à sa section critique. Dans ce cas, la priorité est périodiquement attribuée au processus en état de famine, mais celui-ci, réagissant toujours à contre-temps, n'en profite jamais.

Mettons ce phénomène en évidence dans le cas d'un système formé de trois processus (n = 3) en exhibant une trace d'exécution dans laquelle le processus P_0 est en état de famine (la généralisation à $n \geqslant 3$ quelconque et à un processus quelconque parmi les n processus ne présente aucune difficulté).

Pour cela, définissons, pour le processus P_0 les séquences d'exécution suivantes :

u_0 : exécution de ($ETAT_0$:= en_attente),

v_0 : évaluation de (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors}).

Définissons, d'autre part, pour les processus P_j (j = I,2), concurrents de P_0, les séquences d'exécution suivantes :

s_j : exécution de ($ETAT_j$:= en_attente
test favorable de (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors})),

t_j : exécution de ($ETAT_j$:= engagé
test favorable de (personne d'autre dans {engagé})
SC_j
TRANSLATER PRIORITE DE UN_j
$ETAT_j$:= dehors).

Considérons alors l'état instantané E du système défini par :

- le processus P_I est prioritaire ;
- le processus P_0 vient d'entrer dans son protocole : il a terminé u_0 mais n'a pas encore entamé v_0 ;
- les concurrents P_I et P_2 du processus P_0 sont passifs.

Cet état E est accessible à partir de l'état initial. Or, à partir de l'état E, la séquence infinie A^∞ est réalisable. Cette séquence est la répétition infinie de la séquence A suivante :

A : (s_2 ; v_0 ; s_I ; t_2 ; s_2 ; t_I ; t_2).

Au cours de cette séquence infinie, le processus P_0, en état de famine, est infiniment souvent activé, puisqu'il exécute une infinité de fois v_0. L'hypothèse de vitesse non nulle pour tout processus actif est donc bien respectée.

D'autre part, au cours de cette séquence infinie, le processus P_0 est infiniment souvent prioritaire : il l'est à chaque terminaison de t_I. Mais il réagit toujours à contre-temps. En effet, lorsqu'il exécute v_0, il lui apparait toujours que l'ensemble SUPPOSES PRIORITAIRES SUR SOI est l'ensemble {I,2} (à cause de la

seconde occurence de t_2 dans la séquence A) et que le processus P_2, qu'il juge donc prioritaire sur lui, est candidat (à cause de la première occurence de s_2 dans la séquence A). Ainsi, le processus P_0 évalue toujours défavorablement l'expression (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors}).

4.3. Deuxième étape de la dérivation : insertion d'un mécanisme de contrôle

Dans le protocole (3), lorsqu'un processus candidat est en état de famine, il se retrouve une infinité de fois prioritaire mais ne profite jamais de cette priorité. Pour interdire à un tel scénario de se produire, il suffit d'empêcher ses concurrents de lui prendre indéfiniment son tour. Diverses mesures de contrôle peuvent être prises (cf. paragraphe 6). Par exemple, nous pouvons interdire à tout processus "élu" (i.e. ayant évalué favorablement l'expression (personne d'autre dans {engagé})) de profiter de son élection s'il constate que le processus prioritaire est candidat, auquel cas l'élection n'est pas validée et doit avoir à nouveau lieu.

A cette mesure de contrôle de la validité de l'élection correspond un mécanisme de contrôle que nous composons avec le protocole (3) pour obtenir le protocole (4) suivant (variables et initialisations identiques à celles du protocole (3)) :

```
répéter
    répéter
        ETAT_i := en_attente
        attendre (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors})
        ETAT_i := engagé
    jusqu'à (personne d'autre dans {engagé})
jusqu'à (JE SUIS PRIORITAIRE_i ou PRIORITAIRE NON CANDIDAT_i)
SC_i
TRANSLATER PRIORITE DE UN_i
ETAT_i := dehors
```

Ce protocole est dérivé du protocole (3) par insertion du mécanisme de contrôle (répéter ... jusqu'à (JE SUIS PRIORITAIRE$_i$ ou PRIORITAIRE NON CANDIDAT$_i$)). L'évaluation par le processus P_i du prédicat PRIORITAIRE NON CANDIDAT$_i$ rend la valeur vrai si P_i, en consultant la variable ETAT$_k$ du processus P_k qu'il juge prioritaire, lit la valeur dehors.

L'adjonction de ce mécanisme de contrôle permet de faire hériter le protocole (4) des propriétés intéressantes du protocole (3) : exclusion mutuelle sans risque d'interblocage, évolution équitable des relations de priorité. Ici encore,

un processus P_i bénéficiaire de l'exclusion mutuelle en dispose à partir de l'instant de terminaison de l'évaluation favorable de l'expression (personne d'autre dans {engagé}) et ne la relâche que lorsqu'il termine la modification de la variable $ETAT_i$ (affectation de la valeur dehors ou de la valeur en_attente). Il en résulte qu'un processus quelconque, contrôlant la validité de l'élection, dispose alors d'une vision à jour des relations de priorité.

L'adjonction de ce mécanisme permet, de plus, d'obtenir un protocole équitable : un processus candidat quelconque ne peut être réduit à l'état de famine. Dans ce protocole équitable, l'attente d'un processus candidat quelconque est même bornée. Exprimée en nombre de dépassements possibles, cette borne vaut (n-I). Elle peut être atteinte. Nous donnons en Annexe une démonstration de ces résultats.

5. MISES EN OEUVRE

Nous envisageons ici la mise en oeuvre des protocoles abstraits (3) et (4). Les mises en oeuvre proposées conviennent également au germe (2), mais peuvent évidemment être simplifiées puisque, dans ce germe, les relations de priorité sont invariantes. Il s'agit donc, ici, de proposer des représentations adaptées des relations de priorité et de leur évolution.

Considérons d'abord le protocole abstrait du paragraphe 2 (translation pas à pas d'un privilège sur un anneau virtuel). Dans ce protocole, les relations de priorité et leur évolution peuvent être représentées, soit à l'aide d'une variable partagée, soit à l'aide de n variables de communication.

Une première représentation consiste à utiliser la variable partagée PRIORITE à valeurs dans l'ensemble {0,...,n-I} et de valeur initiale nulle. L'expression $JE\ SUIS\ PRIORITAIRE_i$ est alors représentée par l'expression (PRIORITE = i), la procédure $TRANSLATER\ PRIORITE\ DE\ UN_i$ étant, quant à elle, représentée par l'instruction (PRIORITE := PRIORITE ⊞ I).

Une seconde représentation, nous intéressant plus particulièrement dans cette étude puisqu'elle est distribuée, consiste à utiliser n variables de communication représentant modulo n des compteurs de (Robert 77). En effet, à tout instant et pour tout entier i de l'ensemble {0,...,n-I} :

$JE\ SUIS\ PRIORITAIRE_i$ <==>

$(\Sigma_{j=0,\ldots,n-I}(\text{nombre (terminaisons (}TRANSLATER\ PRIORITE\ DE\ UN_j\text{))))}$ modulo n = i.

Dotons alors chaque processus P_i d'une variable de communication $NTPI_i$ de valeur initiale nulle, variable représentant le compteur (nombre (terminaisons ($TRANSLATER\ PRIORITE\ DE\ UN_i$))) et servant donc à comptabiliser le nombre de translations d'un pas des relations de priorité effectuées par le processus P_i. Il est facile de voir que ces variables $NTPI_i$ peuvent évoluer modulo n et donc prendre leur valeur dans le sous-ensemble {0,...,n-I} de ℕ. Avec une telle représentation :

- l'expression JE SUIS PRIORITAIRE$_i$ est représentée par l'expression ($(\Sigma_{j=0,\ldots,n-I}\ NTPI_j)$ modulo n = i) ;
-la procédure TRANSLATER PRIORITE DE UN$_i$ est représentée par l'instruction ($NTPI_i$:= $NTPI_i \boxplus I$).

Notons que cette mise en oeuvre distribuée du protocole abstrait du paragraphe 2 nous permet d'obtenir l'un des protocoles donnés dans (Raynal 85b). D'autres choix de représentation permettent de retrouver le protocole de (Dijkstra 74) et le principal protocole de (Raynal 85b) (voir (Verjus 86)).

Nous pouvons, sans problème, adopter pour les protocoles (3) et (4) la représentation à l'aide de la variable PRIORITE ou celle utilisant les variables $NTPI_i$.

6. UN ENSEMBLE DE PROTOCOLES D'EXCLUSION MUTUELLE

A partir du germe (2) et sur le thème du protocole (4), nous pouvons nous livrer à diverses variations.

Ces variations jouent sur deux registres :
- à l'alinéa 6.I, sur le choix du mode de translation des relations de priorité,
- à l'alinéa 6.2, sur le choix du mécanisme de contrôle.

Nous présentons, à l'alinéa 6.3, quelques résultats relatifs à ces variations.

De plus, un mécanisme de contrôle étant choisi, nous indiquons à l'alinéa 6.4, à travers un exemple, que l'on peut énoncer des conditions suffisantes d'équité sur le choix du mode de translation pour obtenir des protocoles équitables.

A l'alinéa 6.5, nous donnons une expression formelle du thème précédemment développé. C'est la forme générale des protocoles d'exclusion mutuelle dérivés du germe (2) par translation des relations de priorité et adjonction d'un mécanisme de contrôle. Nous évoquons aussi la possibilité de variations sur le thème du protocole (3) ne comportant aucun mécanisme de contrôle.

6.I. Variations en jouant sur le choix du mode de translation

6.I.I. Translations quelconques

Au lieu de s'en tenir à un mode de translation pas à pas des relations de priorité, nous pouvons envisager un mode quelconque de translation le long de l'anneau virtuel. Nous généralisons alors la procédure TRANSLATER PRIORITE DE UN$_i$ (notée désormais T_{Ii}) en une procédure TRANSLATER PRIORITE$_i$. Le seul effet

de l'exécution de cette procédure est de rendre prioritaire l'un des processus du système en translatant par là-même l'ensemble des relations de priorité.

6.I.2. Exemples

La procédure TRANSLATER PRIORITE$_i$ peut être, par exemple, outre T_{Ii} :

- la procédure T_{2i} : PRIORISER LE SUCCESSEUR$_i$ dont l'effet de l'exécution (par P_i) est de rendre prioritaire le processus $P_{i \boxplus I}$:
  - dans le cadre non distribué évoqué :
    PRIORISER LE SUCCESSEUR$_i$ = (PRIORITE := i $\boxplus$ I) ;
  - dans le cadre distribué évoqué : PRIORISER LE SUCCESSEUR$_i$
    = (NTPI$_i$:= NTPI$_i$ $\boxplus$ ((i $\boxplus$ I) $\boxminus \Sigma_{j=0\ldots n-I}$ NTPI$_j$)).
- la procédure T_{3i} : PRIORISER PREMIER SUCCESSEUR CANDIDAT OU A DEFAUT MOI-MEME$_i$ dont l'effet de l'exécution est de prioriser un certain processus $P_{i'}$; pour chaque exécution, le numéro d'identification i' est calculé par le processus P_i en exécutant :

  J_i := i $\boxplus$ I
  tant que ($J_i \neq$ i et ETAT$_{Ji}$ = dehors) faire J_i := J_i $\boxplus$ I ftq
  i' := J_i

où J_i est une variable locale au processus P_i à valeurs dans l'ensemble {0,...,n-I} et de valeur initiale quelconque ; la procédure T_{2i} est représentée, outre le calcul de i' par :
  - (PRIORITE := i') dans le cadre non distribué,
  - (NTPI$_i$:= NTPI$_i$ $\boxplus$ (i' $\boxminus \Sigma_{j=0\ldots n-I}$ NTPI$_j$)) dans le cadre distribué.
- la procédure T_{4i} : PRIORITE ALEATOIRE$_i$ dont l'effet de l'exécution est de prioriser un certain processus $P_{i'}$; pour chaque exécution, le numéro d'identification i' est obtenu par une évaluation de la fonction aléatoire ALEA$_i$ rendant, indépendamment de l'histoire du système, une valeur i' quelconque de l'ensemble {0,...,n-I} avec une probabilité constante et non nulle ; en dehors du calcul du numéro i', la procédure T_{4i} est représentée de la même manière que la procédure T_{3i}.

Il est à noter que :

- dans le cadre non distribué : la procédure T_{2i} (PRIORITE := i $\boxplus$ I) ne demande qu'une mise à jour de la variable PRIORITE sans consultation préalable de sa valeur (contrairement à la procédure T_{Ii}), n'impose l'exécution d'aucun programme supplémentaire (contrairement aux procédures T_{3i} et T_{4i}) et n'implique aucune communication entre processus (contrairement à T_{3i}) ;

- dans le cadre distribué : la procédure T_{Ii} ($NTPI_i := NTPI_i \oplus I$) est, parmi les quatre modes de translation envisagés, la seule qui n'implique aucune communication entre processus.

Remarquons enfin que, dans un cadre distribué dans lequel les processus communiquent par échanges de messages, (Ricart 83) utilise la procédure T_{3i} (PRIORISER PREMIER SUCCESSEUR CANDIDAT OU A DEFAUT MOI-MEME$_i$) pour obtenir un protocole d'exclusion mutuelle équitable.

6.2. Variations en jouant sur le choix du mécanisme de contrôle

Pour passer du protocole inéquitable (3) au protocole équitable (4), nous avons adjoint au premier un mécanisme de contrôle adapté, appelé mécanisme de contrôle de la validité de l'élection et désormais noté M_I.

Nous aurions pu choisir d'adjoindre au protocole (3), non pas le mécanisme M_I, mais l'un des mécanismes M_2 ou M_3 suivants.

Le mécanisme M_2, plus contraignant que le mécanisme M_I et appelé mécanisme de contrôle renforcé de la validité de l'élection, correspond à la mesure suivante. Cette mesure vise à interdire à tout processus élu de profiter de son élection s'il juge qu'un processus prioritaire sur lui (et non pas seulement le processus prioritaire) est candidat, auquel cas l'élection est invalidée et doit avoir à nouveau lieu. La composition de ce mécanisme avec le protocole (3) permet d'obtenir le protocole (5) suivant (variables et initialisations identiques à celles du protocole (3)) :

répéter
  répéter
    $ETAT_i$:= en_attente
    attendre (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors})
    $ETAT_i$:= engagé
  jusqu'à (personne d'autre dans {engagé})
jusqu'à (tous de PRIORITAIRES SUR SOI dans {dehors})
SC_i
TRANSLATER PRIORITE DE UN_i
$ETAT_i$:= dehors

L'adjonction du mécanisme M_2 (répéter ... jusqu'à (tous de PRIORITAIRES SUR SOI dans {dehors}) permet d'obtenir un protocole équitable en conservant la propriété d'exclusion mutuelle du protocole (3). Plus précisément, ici encore, un processus P_i bénéficiaire de l'exclusion mutuelle en dispose à partir de l'instant de terminaison de l'évaluation favorable de l'expression (personne d'autre

dans {engagé}) et ne la relâche qu'en terminant la modification de la variable $ETAT_i$ (par affectation de la valeur dehors ou de la valeur en_attente). Il en résulte qu'un processus quelconque, contrôlant la validité de l'élection, dispose, ici encore, d'une vision à jour des relations de priorité (d'où la référence à l'ensemble PRIORITAIRES SUR SOI plutôt qu'à l'ensemble SUPPOSES PRIORITAIRES SUR SOI dans l'expression du test de contrôle).

Le mécanisme M_3 relève, quant à lui, d'une approche différente. C'est un mécanisme de contrôle de la modification des relations de priorité : la modification de ces relations par le processus sortant de section critique est interdite si ce processus n'est pas prioritaire et qu'il constate que le processus prioritaire est candidat. La composition de ce mécanisme M_2 (si (JE SUIS PRIORITAIRE$_i$ ou PRIORITAIRE NON CANDIDAT$_i$) alors ... fsi) avec le protocole (3) permet d'obtenir le protocole (6) suivant :

répéter
  $ETAT_i$:= en_attente
  attendre (tous de SUPPOSES PRIORITAIRES SUR SOI dans {dehors})
  $ETAT_i$:= engagé
jusqu'à (personne d'autre dans {engagé})
SC_i
si (JE SUIS PRIORITAIRE$_i$ ou PRIORITAIRE NON CANDIDAT$_i$)
  alors TRANSLATER PRIORITE DE UN$_i$
fsi
$ETAT_i$:= dehors

L'adjonction de ce mécanisme M_3 permet d'obtenir un protocole équitable héritant de la propriété d'exclusion mutuelle du protocole (3). Ainsi, ici encore, le processus P_i évaluant l'expression de contrôle dispose d'une vision à jour des relations de priorité.

6.3. Quelques résultats concernant ces variations

Notons tout d'abord que, quel que soit le choix du mécanisme de contrôle dans l'ensemble $M = (M_1, M_2, M_3)$ et quel que soit le mode de translation choisi dans l'ensemble $T = (T_{1i}, T_{2i}, T_{3i}, T_{4i})$, le protocole obtenu est équitable (voir (Thoraval 86)). En fait, dans le cas de la procédure T_{4i} - attribution aléatoire de la priorité - nous pouvons seulement affirmer que tout processus candidat à sa section critique y accède au bout d'un temps fini avec probabilité un ; l'attente d'un processus candidat quelconque n'est donc pas bornée. Dans le cas des procédures T_{1i}, T_{2i}, T_{3i}, l'attente d'un processus candidat quelconque est tou-

jours bornée.

Il est bien sûr trivial d'obtenir des protocoles inéquitables, par exemple, en choisissant pour la procédure TRANSLATER PRIORITE$_i$ l'une des procédures suivantes : SE DONNER LA PRIORITE$_i$, NE RIEN FAIRE$_i$.

Nous présentons, sous forme d'un tableau, des résultats relatifs à l'attente maximale. Les lignes correspondent à l'un des trois mécanismes de contrôle. Les colonnes à l'un des modes de translation T_{Ii}, T_{2i} ou T_{3i}. Dans chaque case, nous indiquons, pour le protocole résultant, la borne correspondante lorsque nous l'avons établie, ou à défaut, un minorant de cette borne (cf. (Thoraval 86)). Dans les cases de la première colonne -translation pas à pas - nous mentionnons, de plus, pour mémoire, le numéro du protocole correspondant, parmi les protocoles (4), (5) et (6).

	T_{Ii}	T_{2i}	T_{3i}
M_I	(4) $n - I$	$> n - I$	$n - I$
M_2	(5) $n - I$	$n - I$	$n - I$
M_3	(6) $\frac{n\,(n - I)}{2}$	$> \frac{n\,(n - I)}{2}$	$\frac{n\,(n - I)}{2}$

La simple lecture de ce tableau nous permet de constater, par exemple, que l'attente maximale d'un processus candidat quelconque est de (n-I) tours que nous choisissions le couple (M_I, T_{Ii}) ou le couple (M_I, T_{3i}). Il s'avère donc préférable, dans le cas d'une mise en oeuvre distribuée, de choisir le premier couple puisque, contrairement au mode de translation T_{3i}, le mode T_{Ii} n'implique aucune communication entre processus.

6.4. Conditions suffisantes d'équité

Un mécanisme de contrôle étant choisi, il est possible d'énoncer des con-

ditions suffisantes sur le choix du mode de translation qui permettent, quand elles sont remplies, de dériver un protocole équitable voire un protocole offrant une certaine "qualité" d'équité (cf. (Thoraval 86)).

Par exemple, lorsque le mécanisme M_2 de contrôle de la validité de l'élection est retenu, nous pouvons énoncer, de manière informelle, les conditions suffisantes suivantes :

- équité : si la procédure TRANSLATER PRIORITE$_i$ est telle que tout processus candidat devienne prioritaire après un nombre fini d'exécutions de celle-ci, alors le protocole abstrait résultant est équitable (condition vérifiée par T_{Ii}, T_{3i} et T_{4i} (avec probabilité un) mais non vérifiée par T_{2i}) ;
- linéarité de l'attente maximale (en fonction du nombre n de processus) : si la procédure TRANSLATER PRIORITE$_i$ est telle que tout processus candidat devienne prioritaire après au plus (n-I) exécutions de celle-ci survenant après sa déclaration de candidature, alors dans le protocole abstrait résultant, l'attente maximale d'un processus candidat quelconque est de (n-I) tours (condition vérifiée par T_{Ii} et T_{3i}).

Sur la base de telles conditions, il est très facile de dériver de façon fiable des protocoles équitables à partir du germe (2) : la construction de protocoles équitables peut être menée naturellement de pair avec leur preuve.

6.5. Forme générale

Nous pouvons maintenant donner la forme générale des protocoles d'exclusion mutuelle dérivés du germe (2) par translation contrôlée des relations de priorité le long d'un anneau virtuel :

$$2(t, m, r)$$

où le caractère "2" est utilisé par référence au germe (2), où le paramètre t (pour "translation") est à valeurs dans l'ensemble T, le paramètre m (pour "mécanisme de contrôle") est à valeurs dans l'ensemble M, le paramètre r (pour "représentation") est à valeurs dans l'ensemble (R_I, R_2) (R_I correspondant à la représentation non distribuée, R_2 à la représentation distribuée). Notons qu'il est possible d'étendre chacun de ces ensembles.

Cette forme générale nous permet de mettre en évidence, de façon immédiate, les ressemblances et différences existant entre les solutions classiques de (de Bruijn 68), (Eisenberg 72) et (Knuth 66). Ainsi, le protocole de (de Bruijn 68) est $2(T_{Ii}, M_3, R_I)$, le protocole de (Eisenberg 72) étant, à un détail inessentiel près, $2(T_{3i}, M_I, R_I)$. Si nous incluons le mécanisme vide $\emptyset$ dans l'ensemble M, le protocole de (Knuth 66) est alors, à un détail inessentiel près, $2(T_{2i}, \emptyset, R_I)$.

Tout comme le protocole (3), le protocole $2(T_{2i}, \emptyset, R_I)$ ne comporte donc pas de mécanisme de contrôle. Alors que le premier n'est pas équitable, le second l'est (avec, certes, une attente maximale de $(2^{n-I}-I)$ tours ! -cf. (Thoraval 86)-). Il est bien sûr possible de se livrer, sur le thème du protocole (3), à des variations analogues aux précédentes. Mais il n'est pas sûr que les résultats relatifs à l'équité soient alors indépendants du mode de représentation choisi (R_I ou R_2). Nous ignorons, par exemple, si le protocole $2(T_{2i}, \emptyset, R_2)$ est équitable i.e. nous ignorons si la non-atomicité de l'évaluation de l'expression $(\Sigma_{j=0...n-I} NTPI_j)$ nécessaire à l'évaluation de l'ensemble SUPPOSES PRIORITAIRES SUR SOI entraîne, dans ce cas, en l'absence de mécanisme de contrôle, des distorsions désagréables.

7. CONCLUSION

7.I. Le problème de l'exclusion mutuelle

Nous venons de mettre en évidence :

- le mécanisme assurant l'exclusion mutuelle dans les solutions de (Knuth 66), (de Bruijn 68) et (Eisenberg 72) ; il est également utilisé (cf. Annexe) dans celles de (Dijkstra 65), (Katseff 78), (Burns 8I) et (Pnueli 84) ;
- le mécanisme (germe (2)) prévenant tout risque d'interblocage dans les solutions de (Knuth 66), (de Bruijn 68) et (Eisenberg 72) ; il est également utilisé dans celle de (Katseff 78).

Nous avons montré également comment l'équité est assurée dans les solutions de (de Bruijn 68) et (Eisenberg 72). Comme dans la solution de (Dijkstra 74), correspondant à une mise en oeuvre possible du protocole abstrait du paragraphe 2, l'idée de base consiste à faire évoluer les relations de priorité entre processus le long d'un anneau virtuel (idée également adoptée par (Knuth 66)).

Au-delà des solutions classiques de de Bruijn et Eisenberg, solutions non distribuées, nous avons montré comment il est possible de se livrer à diverses variations sur le thème commun à ces solutions et d'obtenir ainsi des protocoles d'exclusion mutuelle équitables, distribués ou non.

La forme générale des protocoles d'exclusion mutuelle dérivés du germe (2) par translation (contrôlée ou non) des relations de priorité sur un anneau virtuel, nous a permis de mettre en évidence, de façon synthétique, les ressemblances et différences entre trois solutions classiques. Ces solutions ((Knuth 66), (de Bruijn 68), (Eisenberg 72)) reposent sur le même germe distribué (2), partagent la même représentation non distribuée des relations de priorité, et ne diffèrent que par le choix du mode de translation et du mécanisme de contrôle (absent dans (Knuth 66)).

Nous avons donné des abstractions des différents mécanismes de synchronisation utilisés dans les solutions classiques (exemples : TRANSLATER PRIORITE$_i$, JE SUIS PRIORITAIRE$_i$, PRIORITAIRE NON CANDIDAT$_i$), ce qui permet :

- d'une part, de mieux comprendre ces mécanismes et ces solutions, en faisant justement abstraction des considérations de mises en oeuvre, et de s'assurer de leur bon fonctionnement à ce niveau abstrait ;
- d'autre part, d'envisager, ensuite seulement, la conception de mises en oeuvre cohérentes, en choisissant telle ou telle représentation distribuée ou non des abstractions utilisées.

Notons enfin que les mécanismes permettant de dériver du germe des protocoles équitables, dépendent de ce germe au sens où ils partagent avec lui des variables communes.

Il est toutefois possible de dériver de ce germe d'autres protocoles équitables en le composant avec des mécanismes d'équité, cette fois, indépendants de lui. Il existe différentes possibilités. Nous en présentons une dans (Thoraval 86), permettant d'obtenir plusieurs variantes de la solution distribuée de (Katseff 78), solution que nous avons, par la même occasion, modifiée pour lui permettre de réaliser une propriété d'équité qu'elle est, à tort, supposée vérifier. De tels mécanismes d'équité, indépendants, en fait, de tout germe, peuvent être composés avec n'importe quel protocole d'exclusion mutuelle sans risque d'interblocage, mais inéquitable, pour en dériver un protocole équitable.

7.2. Au-delà du seul champ de l'exclusion mutuelle

La méthode de construction de protocoles d'exclusion mutuelle que nous avons employée consiste à dériver, à partir d'un protocole très simple mais non satisfaisant (ici le protocole (I)), des protocoles adéquats par une succession de transformations élémentaires. Chaque transformation permet de passer d'un protocole (A) à un protocole (B), par adjonction au protocole (A), d'un mécanisme de synchronisation adapté. Pour chaque transformation, nous justifions, informellement, le choix du mécanisme introduit, en fonction du rôle qu'il doit jouer : doter le protocole (B) d'une propriété absente du protocole (A) tout en le faisant hériter des propriétés intéressantes de ce dernier.

Notre démarche de dérivation demeure informelle. Il est cependant possible de la formaliser, du moins partiellement. Par exemple (voir Annexe), nous pouvons vérifier syntaxiquement que tous les protocoles dérivés du protocole (I) héritent de la propriété d'exclusion mutuelle possédée par celui-ci.

Notre démarche de dérivation permet d'envisager de mener de pair construc-

tion et preuve de protocoles. Elle permet également de mettre en évidence le rôle des différents mécanismes à l'oeuvre dans les protocoles résultants.

Cette méthode de construction nous semble indépendante du mode de communication entre processus : elle peut être utilisée pour construire des protocoles distribués pour lesquels la communication entre processus s'effectue par échanges de messages.

Elle nous semble également applicable à d'autres problèmes, tels ceux de la détection de l'interblocage ou de la terminaison. Elle rejoint d'ailleurs, d'un certain point de vue, la méthode employée par (Francez 81) pour retrouver notamment le protocole de détection de la terminaison de (Francez 79), ou celle utilisée par (Dijkstra 83) pour construire un autre protocole de détection de la terminaison.

BIBLIOGRAPHIE

(André 83) F. André, D. Herman, J.P.Verjus
Synchronisation de programmes parallèles
- expression et mise en oeuvre dans les systèmes centralisés ou distribués -
Dunod Informatique, Paris 1983

(Bochmann 79) G.V. Bochmann
Architecture of distributed computer systems
LNCS 77, Springer Verlag, 1979

(Burns 81) J.E. Burns
Symmetry in systems of asynchronous processes
Proc. of the 22nd Ann. Symp. on Found. of Comp. Sc., oct. 81, pp. 169-174

(de Bruijn 68) N.G. de Bruijn
Additional comments on a problem in concurrent programming control
Comm. of the ACM, 11(1), jan. 1968, pp. 55-56

(Dijkstra 65) E.W. Dijkstra
Solution of a problem in concurrent programming control
Comm. of the ACM, 8(9), sept. 1965, p. 569

(Dijkstra 74) E.W. Dijkstra
Self stabilization in spite of distributed control
Comm. of the ACM, 17(11), nov. 1974

(Dijkstra 83) E.W. Dijkstra, W.H.J. Feijen, A.J.M. van Gasteren
Derivation of a termination detection algorithm for distributed computations
Inf. Proc. Letters, vol.16, june 1983, pp. 217-219

(Eisenberg 72) M.A. Eisenberg, M.R. Mc Guire
Further comments on Dijkstra's concurrent programming control problem
Comm. of the ACM, 15(11), nov. 1972, p. 999

(Francez 79) N. Francez, M. Rodeh
Achieving distributed termination without freezing
TR-72, IBM Israel Scientific Center, 1979

(Francez 81) N. Francez, M. Rodeh, M. Sintzoff
Distributed termination with interval assertions
Proc. of the Intern. Coll. on Formalization of Programming Concepts,
Peniscola, Spain, april 19-25, 1981

(Katseff 78) H.P. Katseff
A new solution to the critical section problem
Conf. Rec. of the 10th Ann. ACM Symp. on Th. of Comp.,
San Diego, Calif., may 1-3, 1978, pp. 86-88

(Knuth 66) D.E. Knuth
Additional comments on a problem in concurrent programming control
Comm. of the ACM, 9(5), may 1966, pp. 321-322

(Pnueli 84) A. Pnueli, L. Zuck
Verification of multiprocess probabilistic protocols
Proc. of the 3rd ACM SIGACT-SIGOPS Symp. on Princ. of Distr. Comp.,
Vancouver, Canada, aug. 27-29, 1984, pp. 12-17
et également dans
Distributed Computing, Springer Verlag, 1, 1986, pp. 53-72

(Raynal 84) M. Raynal
Algorithmique du parallélisme
- le problème de l'exclusion mutuelle -
Dunod Informatique, Paris 1984

(Raynal 85a) M. Raynal
Algorithmes distribués et protocoles
Eyrolles, Paris 1985

(Raynal 85b) M. Raynal
Un algorithme d'exclusion mutuelle pour une structure logique
en anneau
T.S.I., 4(5), 1985, pp. 471-474

(Ricart 83) G. Ricart, A.K. Agrawala
Author's response to "On mutual exclusion in computer networks" by Carvalho and Roucairol
Comm. of the ACM, 26(2), feb. 1983, pp. 147-148

(Robert 77) P. Robert, J.P. Verjus
Towards autonomous description of synchronization modules
Proc. IFIP Congress, North Holland, 1977, pp. 981-986

(Rosenkrantz 78) D.J. Rosenkrantz, R.E. Stearns, P.M. Lewis II
System level concurrency control for distributed database systems
ACM Trans. on Database Systems, 3(2), june 1978, pp. 178-198

(Thoraval 86) R. Thoraval
Dérivation de protocoles d'exclusion mutuelle équitables à partir de protocoles d'exclusion mutuelle inéquitables
Rapport IRISA n° 304, juin 1986

(Thoraval 87) R. Thoraval
Abstraction et dérivation de protocoles distribués (titre provisoire)
Thèse (en préparation), Rennes, 1987

(Verjus 86) J.P. Verjus, R. Thoraval
Dérivation d'algorithmes distribués d'arbitrage
T.S.I., 5(1), jan.-fev. 1986, pp. 37-47

ANNEXE

I. L'équité dans le protocole (4)

Pour montrer que le protocole (4) est équitable, notons d'abord que, si à un instant t_i quelconque de l'histoire du système :

- le processus P_i est prioritaire (i entier quelconque de l'ensemble {0,...,n-I}) ;
- ce processus P_i est candidat i.e. $ETAT_i \neq$ dehors ;
- aucun processus P_j du système n'est en cours d'évaluation de l'expression ($JE\ SUIS\ PRIORITAIRE_j$ ou $PRIORITAIRE\ NON\ CANDIDAT_j$) ni en cours d'exécution de (SC_j ; $TRANSLATER\ PRIORITE\ DE\ UN_j$) ;

alors, le processus P_i est nécessairement le premier à entrer en section critique après l'instant t_i. En effet, à partir de l'instant t_i et tant qu'il n'est pas entré en section critique, le processus P_i bloque l'accès de chacun de ses concurrents éventuels à leur section critique, pour le moins grâce au test de contrôle de la validité de l'élection, test effectué en exclusion mutuelle (donc avec une vision à jour des relations de priorité). Comme le protocole (4) est sans risque d'interblocage, le processus P_i entre alors fatalement en section critique.

Or, le caractère équitable de l'évolution des relations de priorité assure que pour tout processus candidat P_i ne parvenant pas à entrer en section critique sans être prioritaire, il existe un tel instant t_i postérieurement à sa déclaration de candidature.

L'attente maximale d'un processus candidat quelconque P_i est de (n-I) tours car, si ce processus ne parvient pas à entrer en section critique sans être prioritaire, l'instant t_i survient fatalement après au plus (n-I) translations d'un pas des relations de priorité.

Notons que la démonstration du caractère suffisant des conditions d'équité présentées à l'alinéa 6.4 peut être menée de manière analogue.

2. L'héritage de la propriété d'exclusion mutuelle chez les descendants du protocole (I)

Considérons la forme générale F suivante de protocoles généralisant, en quelque sorte, le protocole (I). Pour tout protocole de forme F :

- chaque processus P_i est doté, pour le moins, d'une variable de communication $ETAT_i$ à valeurs dans un ensemble incluant l'ensemble {dehors, engagé}, variable de valeur initiale dehors ;

- le programme de chaque processus P_i a la forme suivante :

A_i
$ETAT_i$:= engagé
si (quelqu'un d'autre dans {engagé}) alors aller à $Etiquette_i$ fsi
B_i
SC_i
C_i
$ETAT_i$:= dehors

où :

- A_i est une séquence non vide d'instructions sans branchement hors de A_i ;
- $Etiquette_i$ correspond à une instruction de A_i ;
- B_i est une séquence, éventuellement vide, d'instructions sans modification de la variable $ETAT_i$ et ne pouvant comporter de branchements que vers des instructions de A_i ou de B_i ;
- C_i est une séquence, éventuellement vide, d'instructions sans modification de la variable $ETAT_i$ et sans branchement hors de C_i.

En adoptant la même démarche que pour le protocole (I) (cf. alinéa 3.I), nous pouvons établir que tout protocole de forme F assure l'exécution de l'action SC_i (mais également de B_i et C_i) en exclusion mutuelle.

Il est alors facile de vérifier syntaxiquement que tous les protocoles dérivés du protocole (I) assurent l'exécution en exclusion mutuelle de l'action SC_i. Tous ces protocoles sont, en effet, de forme F, tout comme le protocole (I) lui-même.

De plus, le fait que l'exclusion mutuelle porte sur l'ensemble des actions B_i, SC_i et C_i confirme ce que nous avons dit aux paragraphes 4 et 6 concernant la vision des relations de priorité par le processus élu.

Notons que les protocoles de (Dijkstra 65) et (Katseff 78) sont aussi de forme F ; les protocoles de (Burns 81) et (Pnueli 84) sont, quant à eux, d'une forme très voisine de la forme F. Dans tous ces protocoles, l'exclusion mutuelle est garantie de la même façon que dans le protocole (I).

Nous venons de considérer le problème de l'héritage de la propriété d'exclusion mutuelle chez les descendants du protocole (I). D'une façon analogue (voir (Thoraval 87)), nous pouvons régler le problème de l'héritage de la propriété d'exclusion mutuelle sans risque d'interblocage dans la descendance du germe (2). Il suffit de définir, à partir du germe, une forme F' adaptée de protocoles d'exclusion mutuelle sans risque d'interblocage.